6.3 Personnel

The proposed project will be under the direction of Thomas Marill. Coordination with DEC will be handled by Dr. Marill.

The CCA project engineer will be Hallam Murray, who has been project engineer on the experimental network. Assisting Mr. Murray in an advisory capacity will be William F. Mann, CCA Chief Programmer, and George Mealy, consultant to CCA.

Working directly under Mr. Murray will be a team of two full-time programmers and one programmer/technical-writer, who will be in charge of system documentation.

Professor Gordon Bell will act as design consultant to the entire project.

Biographies of Bell, Mann, Marill, Mealy, and Murray follow.

C. Gordon Bell

Mr. Bell holds the S.B. and S.M. degrees (1957) from Massachusetts Institute of Technology. He is currently Associate Prcfessor, Departments of Computer Science and Electrical Engineering, Carnegie-Mellon University. He is engaged in research on computer networks, automatic design, very small time-shared computers, and distributed computers.

From 1960 to 1966, Mr. Bell was on the staff of Digital Equipment Corporation, where, among other projects, he collaborated on the design of the PDP-1 computer, was project leader on the PDP-4, did the system design of the PDP-5, was project leader for the PDP-6, and was Manager of Large Computer Engineering.

A list of Mr. Bell's papers and publications follows.

C. Gordon Bell (cont.)

"Introduction to Time Sharing", Computer Design, February-March 1968.

"Time Shared Computer Bibliography", Proc. IEEE, December 1966.

"Time Shared Computers", ARPA Report, Carnegie Institute of Technology, Pittsburgh, Pennsylvania, May 1967.

"Communications and Computers", Carnegie Review No. 12, July 1967. (Also given orally at Carnegie Conference, April 1967.)

"Symposium on Computers of the 1970s", participant in Fall Joint Computer Conference, November 1964, conducted by Oliver Selfridge.

"The Pulsed Analog System for Evaluating Correlation Functions for Radar", AFCRL-TN-60-963. Scientific Report No. 1(8494-R-1) of the Electronic Systems Laboratory.

"Computer Techniques", published by Instrumentation Techniques in Nuclear Pulse Analysis, National Academy of Sciences, National Research Council, Publication No. 1184.

"A Time-Shared Computer for Real-Time Information Processing", with John Leng, J.A. Quarrington, and P.K. Patwardham, published by Instrumentation Techniques in Nuclear Pulse Analysis, National Academy of Sciences, National Research Council, Publication No. 1184.

"A Case for Organized Input-Output", talk to Washington, D.C. and Boston, Mass., ACM chapters, January-February meeting, 1963.

"Reduction of Speech Spectra by Analysis-by-Synthesis Techniques", with H. Fujisaki, J.M. Heinz, K.N. Stevens, and A.S. House, The Journal of the Acoustical Society of America, Vol. 33, No. 12, pp. 1725-1736, December 1961.

Four oral papers on speech research co-authored with members of the Speech Communications Laboratory, M.I.T., published in the Proc. of the International Congress of Acoustics, Germany, August 1959; Acoustical Society, Ottawa meeting, May 1959; Air Force Symposium on Speech Recognition, Cambridge Air Force Research Center, Sept. 28-30, 1959; and Acoustical Society, Cleveland meeting, Oct. 22, 1959.

William F. Mann

In 1962 Mr. Mann became a member of the technical staff at Bolt Beranek and Newman Inc., where he worked on system programming assignments. In 1965 he joined Computer Corporation of America as chief programmer.

The following represents a partial list of the programming efforts with which Mr. Mann has been associated. On over half of these projects Mr. Mann has had full design and/or implementation responsibility.

1. MACRO Assembly Program and DDT On-Line Debugging System (PDP-1).

2. MIDAS Assembly Program (PDP-1).

3. "Expensive Typewriter" On-Line Text-Editing System (PDP-1).

4. Sequence-Break System Utility Package (PDP-1).

5. Scoring Program for the World Sport Parachuting Championship (PDP-1).

6. LISP System (PDP-6).

7. Original Time-Sharing System (PDP-1).

8. Extended DDT On-Line Debugging System for Use with MIDAS (PDP-1).

9. Floating-Point Simulator (PDP-1).

10. Medication-Order Program for Use with Hospital Time-Sharing System (PDP-1).

11. Drum-Oriented Text-Editor System (PDP-1).

12. Chess-Playing Program (IBM 7094).

13. DDT On-Line Debugging System (CDC 3200).

14. On-Line Engineering Computation Aid (CDC 3200).

15. TV Camera Input Program (PDP-6).

William F. Mann (cont.)

16. Computer-to-Computer Text and Display Communication Package (TX-2).

17. Model 101 Information Retrieval Software Package (IBM 360).

18. Model 103 Information Retrieval Software Package (IBM 360).

Thomas Marill

Dr. Marill received the B.A. degree with high honors from Swarthmore College in 1951, the M.A. degree from Cornell University in 1953, and the Ph.D. degree in Experimental Psychology from the Massachusetts Institute of Technology in 1956.

During his graduate studies at M.I.T., he was on the staff of the Acoustics Laboratory and of the Research Laboratory of Electronics, and was engaged in research on detection theory, mathematical models of sensory processes, and man-machine information transfer. From 1956 to 1958 he was on the staff of the M.I.T. Lincoln Laboratory, where he was concerned with SAGE, the first of the large-scale military computer systems. From 1958 to 1965 he was with Bolt Beranek and Newman Inc., becoming Head of the Information Systems Department in 1961. This group pioneered in the areas of time-sharing, automatic pattern recognition, information retrieval, computer-based educational technology, computer-aided graphics, and automatic film-reading. In 1965 Dr. Marill joined Computer Corporation of America, of which he is a founder.

Dr. Marill is a member of the American Association for the Advancement of Science, the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, the American Psychological Association, and the New York Academy of Sciences. He is a past member of the Editorial Board of the Transactions on Human Factors in Electronics, and was Guest Editor for its special issue on Automation of Human Functions.

Dr. Marill is the inventor (with D.J. Edwards) of "Method and System for Remote-Location Computer Communication via Telephone", U.S. Patent 3,347,988.

A list of Dr. Marill's publications follows.

Thomas Marill (cont.)

"Aural Presentation of Information", (with J.C.R. Licklider and U.R.G. Neisser) Acoustics Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (1954).

"Tilt Adaptation and Figural After-Effects", (with Eric G. Heinemann) J. Exper. Psychol., 48, 468-482 (1954).

"Detection Theory and Psychophysics", Technical Report 319, Research Laboratory of Electronics, Massachusetts Institute of Technology (1956).

"The Psychological Refractory Phase", Brit. J. Psychol., 48, II, 93-97 (1957).

"Summary Report of Experimental SAGE Sector Evaluation: The Identification Function", Technical Memorandum 66-4, M.I.T. Lincoln Laboratory, Lexington, Massachusetts (Secret) (1958).

"Statistical Recognition Functions and the Design of Pattern Recognizers", (with David M. Green), IRE Trans. on Electronic Computers, EC-9, 472-477 (1960).

"Automatic Recognition of Speech", IRE Trans. on Human Factors in Electronics, HFE-2, 34-38 (1961).

"Progress in Artificial Intelligence", (Editorial) IRE Trans. on Human Factors in Electronics, HFE-2, 2 (1961).

"Computational Chains and the Simplification of Computer Programs", IRE Trans. on Electronic Computers, EC-11, 173-180 (1962).

"R.F. Reiss's 'An Abstract Machine Based on Classical Association Psychology'" (Critical Review) IRE Trans. on Electronic Computers, EC-11, 587 (1962).

"General Recognition Processes" Northeast Electronics Research and Engineering Meeting, Boston, Mass., November 1962.

"On the Effectiveness of Receptors for Recognition Systems", (with David M. Green) IEEE Trans. on Information Theory, IT-9, 11-17 (1963).

"PIP: A Photo-Interpretative Program for the Analysis of Spark-Chamber Data", (with H. Rudloe and M. Duetsch), Communications ACM, Vol. 6, No. 6, 332-335 (1963).

Thomas Marill (cont.)

"A Note on Pattern-Recognition Techniques and Game-Playing Programs", Information and Control, Vol. 6, 213-217 (1963).

"CYCLOPS-1: A Second-Generation Recognition System", (with A.K. Hartley, T.G. Evans, B.H. Bloom, D.M.R. Park, T.P. Hart, and D.L. Darley), Proc. AFIPS Fall Joint Computer Conference, 27-33 (1963).

"DATA-DIAL: Two-Way Communication with Computers from Ordinary Dial Telephones", (with D.J. Edwards and W. Feurzeig), Communications ACM, Vol. 6, No. 10, 622-624 (1963).

"CYCLOPS-2: A Computer System that Learns to See", (with B.H. Bloom), Technical Report TR65-RD1, Computer Corporation of America, Cambridge, Massachusetts (1965).

"Learning and Perceptual Processes for Computers", (with B.H. Bloom), Advances in Biomedical Computer Applications, Annals of the New York Academy of Sciences, Vol. 128, Art. 3, 1029-1034 (1966).

"A Cooperative Network of Time-Sharing Computers: Preliminary Study", Technical Report No. 11, Computer Corporation of America, Cambridge, Massachusetts (1966).

"Toward a Cooperative Network of Time-Shared Computers" (with L.G. Roberts), Proc. AFIPS Fall Joint Computer Conference, 425-431, November 1966, San Francisco, California.

"A Model of Visual Scene Analysis", in Models for the Perception of Speech and Visual Form, W. Wathen-Dunn (Ed.), The MIT Press, Cambridge, Massachusetts (1967).

George H. Mealy

Mr. Mealy received the A.B. degree from Harvard College in 1951.

From 1951 to 1959, he was with Bell Telephone Laboratories, Inc. where he worked in logical design, switching theory, and systems programming. In 1957, he constructed one of the early operating systems for the IBM 704 computer and the initial IBM 704 implementation of IPL-V. From 1959 to 1963, he was with the RAND Corporation, where he managed the system programming group and did extensive work on the SHARE Operating System for the IBM 709 computer and work in programming research. From 1963 to 1967, he was with the IBM Corporation. He participated in the planning and design of Operating System/360, followed by work in the Systems Architecture area. Since early in 1967 he has been a private consultant.

Mr. Mealy has participated heavily in the SHARE organization since 1957, serving on its Executive Board during 1960-1962, and is currently assistant manager of the SHARE Systems Division. He is also a member of the Association for Computing Machinery.

A list of Mr. Mealy's publications follows.

"A Method for Synthesizing Sequential Circuits", BSTJ, September 1955.

"Operating Systems", the RAND Corporation Paper P-2584, May 1962.

"Anatomy of an Assembly System", The RAND Corporation Paper P-2674, December 1962.

"A Generalized Assembly System", The RAND Corporation Research Memorandum RM-3646-PR, August 1963.

"Information Processing Language V Manual", Second Edition, Prentice-Hall, 1963 (joint author).

"Another Look at Data", Proc. AFIPS 1967 Fall Joint Computer Conference, Thompson, 1967, 525-534.

Hallam G. Murray

Hallam G. Murray received the S.B. degree in Electrical Engineering from Massachusetts Institute of Technology in June 1965. His undergraduate thesis was on ultra-fast linear transforms by computer.

While an undergraduate, he worked at the M.I.T. Instrumentation Laboratory on the check-out of electric circuits, and at the Engineering Standards Laboratory and the Advanced Signal Processing Section of General Electric. In the latter position, he wrote computer programs for plotting and optimizing the ambiguity functions of radar waveforms.

After graduation, Mr. Murray joined the research staff of the M.I.T. Biology Department, where he was concerned with computer manipulation and display of complex molecular structures. He also wrote the communication package which allows the PDP-7 at the Biology Department to communicate directly over a high-speed data-phone link to the Project MAC computer, allowing these two computers to interchange information.

Mr. Murray joined Computer Corporation of America in May 1966. At CCA he has been the Project Engineer on the Experimental Computer Network, a project for tying together by communication lines three computers, one on the West Coast, one in the Boston area, and one in Washington, D.C. This network represents one of the first implementations in which the computers themselves can call each other on an as-needed basis.

The effects of this routing doctrine are several:

1. Transmission is always in the direction of the terminating point--cyclic routing cannot occur.

2. The cost of providing direct routes between nodes is taken into account, depending on the traffic volumes and distances involved.

3. The routing tables change only when a change in network configuration takes place.

With regard to the last point, taking a node or link out of service is handled by marking the link or links at neighboring nodes in the network busy during the outage. The routing tables are not changed, however, in response to a temporary outage.

For the case of the ARPA model, link occupancies (i.e., proportion of time a link is in use) do not contribute significant traffic delays. In the case of the installed network, any attempt to estimate network capacity and delays on the basis of the end-to-end estimates given in Appendix E of the ARPA specifications might be highly misleading. Instead, it is proposed that a calculation be made based on actual experience with the test network and that the calculations be revised as additional nodes are added to the network and further operating experience is gained.

3.6 Node and Network Capacity and Delays

To obtain maximum HOST-IMP input rate at a node with all other nodes in the network quiet, we must first compute the allowable message delay attributable to CPU queuing. The delays due to transmission, modems, and communications over a three link route, HOST to HOST, amount to 46.1 ms. The allowable delay per IMP is thus 112.5 ms, if total delay is to be held to 0.5 second.

For this purpose, we account for memory interference due to packet transmission; this swells the CPU effective processing time by 0.5 ms. The total CPU holding time, h, is given in the following table for CPU processing times of one millisecond. The CPU occupancy is assumed to be equal to one (it will be greater than 0.98 when the allowable maximum delay is reached, according to the queuing curves used). Hence, the capacity in packets per second is the reciprocal of the effective holding time. It is given below; the third column gives the effective input rate in kilobauds.

h	λ (packets/sec.)	I(KB)
1.5	676	2420

To compute the average message delay for the ARPA model, we again use the effective CPU delay for the submitted load of 479 KB., tabulating the input load, effective holding time, CPU occupancy, average CPU delay, and total message delay.

λ	h	α	$\bar{t}$	D
479	1.5	0.72	3.76	67.1

3.7 Discussion

As stated in the Introduction to this chapter, the queuing model employed may be expected to give pessimistic results for capacity and delays. The assumptions made are inaccurate in directions which have the following effects:

1. Service in random order is assumed. The other extreme assumption would be service in FIFO order. The effect on the average delays is to make them conservative by roughly a factor of two; service will normally be in FIFO order, except for packets retransmitted due to transmission errors or IMP overload peaks.

2. Random demand for service is assumed, from infinite sources. In fact, the variance of the interarrival times (between demands for service) will be smaller, and only a finite number of sources of demand exist. The effect, in both cases, is to make the predicted delays somewhat larger than they should be.

3. The small chance of transmission of a packet when no buffer is available at the receiving end will result in a small increase in effective traffic load due to the necessity of retransmission. This effect can be minimized if the IMP sends hold messages to one or more of its neighbors during traffic peaks, at the cost of increasing the delays encountered for held packets.

In the calculations performed above, it has been seen that the critical parameter in the case of the ARPA model is CPU processing

time. In the installed network, delays due to overloaded groups of links may also become significant, depending on the actual network configuration and traffic loads submitted to the network. As pointed out in Section 3.5, reliable estimates for the installed network cannot be made at this time.

Chapter 4
System Implementation Procedure and Schedule

4.1 Introduction

There are six systems involved in the implementation. System I is a PDP-8I available at CCA for the initial programming work. System T is a test system that can perform the functions of both HOST and IMP and is used to check out each IMP and the entire network. T is used in lieu of HOST at each installation prior to turning over the delivered IMP to the HOST programmers. Systems 1, 2, 3, and 4 are the IMPs that form the initial network.

Only systems 1, 2, 3, and 4 are delivered as part of the proposed contract.

Systems 1, 2, 3, and 4 have 12K core memories. System T has 16K. System I has 8K.

4.2 Implementation Schedule

Month 1: Software design and implementation begins.
Order PDP-8I for systems 1, I, and T.
Order logic for systems 1 and T.

Month 2: Order PDP-8I for systems 2 and 3.
Order logic for systems 2 and 3.
System I delivered at CCA.

Month 3: Receive PDP-8I for systems 1 and T.
Receive logic for systems 1 and T.
Assemble systems 1 and T.
Order PDP-8I for system 4.
Order logic for system 4.

Month 4: Start check-out of systems 1 and T.
Receive PDP-8I for systems 2 and 3.
Receive logic for systems 2 and 3.
Assemble systems 2 and 3.

Month 5: Finish check-out of systems 1 and T.
Start check-out of systems 2 and 3.
Receive PDP-8I for system 4.
Receive logic for system 4.
Assemble system 4.

Month 6: IMP software ready.
Tester software ready.
Finish check-out of systems 2 and 3.
Start test of hardware and software using systems 1 and T.
Start check-out of system 4.

Month 7: Test hardware and software using systems 1, 2, 3, and T.
Finish check-out of system 4.

Month 8: Network check-out using 1, 2, 3, 4, and T.

Month 9: Deliver systems 1, 2, 3, and 4.

Month 10-12: Check-out hardware/software system in the field.

Chapter 5
DEC Qualifications and Personnel

5.1 Special Systems

DEC is well known for its interest in working with clients to develop systems designed to meet unique requirements. In the case of the IMP project, DEC proposes to use its experience, gained over many years, to develop a total system geared to ARPA's special needs; since DEC management takes a very direct interest in this project*, the company intends to use all those technical resources which are necessary to assure the success of this project.

In particular, DEC plans to draw heavily on the experience of its Computer Special Systems Group. This group produces, on the average, 15 special systems per month, each requiring a close relationship with the client to establish total system hardware and software specifications, custom engineering to design and implement the system, and follow-through to assure a fully operational system.

A few examples of special systems produced by DEC are as follows.

1. Data concentrators for use with time-sharing systems (ITT, Applied Logic, Badger Meter, others).

---

* See President's letter, page i.

2. Communication concentrators for educational applications (RCA, others).

3. Synchronous modem interfaces (Bell Telephone Laboratories, University of Michigan, M.I.T., others).

4. Industrial data acquisition and control systems (Shell Oil, Dow Chemical, Esso, Pittsburgh Plate Glass, others).

5. High energy physics (University of California, M.I.T., Yale, University of Michigan, others).

6. Graphic systems (Sandia, Boeing, M.I.T., others).

5.2 Product Line Geared to Communications

The application of computers to data communications is one of DEC's foremost specialities. In 1962, DEC built for ITT the world's first computer-based store and forward switching system. The system was designed around the PDP-1 (the world's first commercially available small general-purpose computer). So successful was this first system that 12 additional ones were delivered to ITT for installation all over the world, and are still serving the users.

The more recent PDP-8 has found many applications in the communications area. Today, there are nearly 150 PDP-8's tied to synchronous modems, serving applications such as remote data entry terminals, remote line concentrators, and remote graphic displays.

One of DEC's most popular products in the area of data communication is the 680 Data Communications System which uses the PDP-8 as the control unit. This system has an incremental line cost of $50.00, lower than any other manufacturer on the market. To date,

DEC has delivered over 100 of these systems for use in many applications over the world. So acceptable was the price, performance, and reliability that a number of computer manufacturers are using DEC 680 systems as the solution to their communication problems.

Some typical 680 and PDP-8 applications are:

- Front end to IBM/360
- Front end to Burroughs 5500
- Front end to Univac 1108
- Front end to CDC 3000 series.

5.3 Personnel

Nick J. Mazzarese, Vice President in charge of small computers, will be in charge of the overall project. The project engineer will be John H. Holland, a senior project engineer in the Computer Systems Group. Other staff members involved will be Richard Tringale and Kenneth Brabitz. Biographies of those named above follow.

Kenneth E. Brabitz

EDUCATIONAL BACKGROUND:

1962-1964 U.S. Army
Army Security Agency
Training Fort Devens, Massachusetts

U.S. Army Signal School
Fort Monmouth, New Jersey

Top Secret Crypto Clearance, primary duties included maintenance and repeat of Crypto equipment in Berlin, Germany.

1960-1961
Northeastern University
Boston, Massachusetts
Engineering courses for 1.5 years

1954-1958
Graduate--Hempfield High School
Landisville, Pennsylvania

EMPLOYMENT HISTORY

1966 to Present
DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts
Position: Senior Technician

Currently employed at Digital Equipment Corporation in the Computer Special Systems group. Duties include design and checkout of customer required equipment specializing in data communications systems, project responsibility involving various message concentrator systems and medium speed modem interfaces connected with Bell 201, 203 and 205 modems, high-speed modem interfaces connected to Bell 301 and 303 modems, assisted in the design and implementation of PDP to IBM 360, 7040, 7044 and 7090 computers.

1965-1966
HONEYWELL INCORPORATED
Position: Technician

Special Systems group, participated in the development of standard peripheral equipment including magnetic tape equipment, drums and data communications equipment.

John Henry Holland

EDUCATIONAL BACKGROUND:

1954-1959

Undergraduate work for Bachelor of Applied Science, University of British Columbia. Degree from Electrical Engineering Department.

1959-1962

Graduate work for Master of Applied Science, University of British Columbia. Degree from Electrical Engineering Department.

AWARDS:

Session 1957-1958

Winner of the B.C. Transformer Co., Ltd. scholarship in Electrical Engineering (proficiency in third year).

May 1959

National Research Council Bursary.

May 1960

National Research Council Studentship.

PROFESSIONAL EXPERIENCE:

January 1963-July 1967

CANADIAN ARMAMENT RESEARCH AND DEVELOPMENT ESTABLISHMENT
Valcartier, Quebec, Canada

August 1967 to Present

DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts

January 1963-December 1963

Conducted studies using low-light electro-optical techniques including image intensification and television equipment.

January 1964 to April 1966

Designed and built electronic circuitry used to control the operation of an experimental Q-switched laser (ruby rod) rangefinder. The rangefinder was used as one component of a fire control system which used a DEC PDP-5 computer, a

small special purpose analog computer, and a small dc servo system. Designed and built the interface for the PDP-5 to input data from the rangefinder and output data through D/A converters to the analog computer. The interface also allowed the PDP-5 to monitor all important parameters of the experiment through an A/D converter and multiplexer. The PDP-5 was programmed to simulate the operation of the proposed hard-wired computers. Program responsibility included most of the programs (occupying about 3K of the 4K PDP-5 memory) for the system and had responsibility for field trials and demonstrations of the total fire control system.

May 1966 to July 1967

Project responsibility was for the hardware design and programming for a computer controlled photographic film reader. The system consisted of a DEC PDP-5 computer, an ASR 33 teletype, a Ferranti-Packard high-speed paper tape reader, a Soroban high-speed paper tape punch, a DEC type 30N display, a 503 Tektronix oscilloscope, an Optical Mechanical Unit (film handler and optical equipment) from Information International Inc., and a rack of interface logic. The system can be used to read almost any type of information recorded on film.

During this period involvement with other projects on a short-term or part-time basis included: consulting on the construction of a spectrum analyzer using a PDP-8/S to do integrations of the FM records of absorption spectra of the upper atmosphere: designing and building a dual A/D converter and additional control logic to digitally record on film the output of a hot-wire anemometer used in the wake studies of high velocity projectiles; designing and building a CRT display console for use in the simulation of war games. The system used a PDP-8 with 8K of memory, an ITT CRT and a DEC type 370 light pen. The interface to the display used a single-cycle data break of the PDP-8 and had special features to allow "blink" control, edge detection, variable intensity and special skip facilities. Consulting with the design of a depth charge fuze tester using a PDP-8 and DEC logic. The tester was designed to automatically test the fuze for circuit continuity and for magnetometer sensitivity. Served on a committee for the selection of new computing facilities at CARDE that resulted in the selection of a SIGMA-7 for the new digital computer and EAI 8800/640 for the hybrid facility.

August 1967 to Present

Project engineer in the Computer Special Systems Group at DEC. The first design responsibility was an addressing modification to a PDP-9 to allow memory mapping. Design for a block-buffer for driving D/A converters on the data channel of the PDP-9. Project engineer for a PDP-8/S Data Acquisition system for use on Navy submarines. Project responsibility on an industrial data acquisition and output control system for a PDP-10. The system consists of an integrating digital voltmeter for analog input signals, a contact interrogation unit for scanning digital input signals, and a high power output driver buffer for computer controlled output. Because time for system checkout on a PDP-10 was limited, a PDP-10 I/O bus simulator using a PDP-8 was built. Included in this responsibility was a complete set of diagnostics for the system on the PDP-8. Project engineer on a PDP-9T (the PDP-9T is a PDP-9 which has been modified to permit real-time control of laboratory apparatus in a time-shared environment) is a current project. The PDP-9T has memory mapping to give protection to each user and also traps certain illegal user instructions.

PUBLICATIONS AND PRESENTATIONS:

J.H. Holland:
"A Hybrid-Coupled Tunnel-Diode Amplifier", M.A. Sc. Thesis, University of British Columbia, 1963.

J.H. Holland:
"Control Circuitry for a Laser Rangefinder", C.A.R.D.E. T.N. 1679/65, 1965.

W.G. Thistle and J.H. Holland:
"Analogue Circuitry for Rapidaim", C.A.R.D.E. T.N. 1681/65, CONFIDENTIAL.

W.G. Thistle, J.H. Holland, and J. Higgins:
"RAPIDAIM--A Sighting Aid for Anti-Tank Weapons, Part II--Design and Test of Experimental Model", C.A.R.D.E. T.R. 535, 1965, CONFIDENTIAL.

J.H. Holland and G.T. Pullan:
"Night Trials of Rapidaim", C.A.R.D.E. T.N. 1725/66, 1966, CONFIDENTIAL.

J.H. Holland and D.G. Galbraith:
"Computer-Controlled Film Reader", Paper presented at D.R.B. Data Processing Symposium at Suffield, Alberta, Canada, Spring, 1967.

Nick J. Mazzarese

EDUCATIONAL BACKGROUND:

B.S. M.E., Stevens Institute of Technology, 1955.
M.S. E.E., Northeastern University, 1961.

PROFESSIONAL EXPERIENCE:

1961 to Present
DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts

Vice President, Group Manager
Small Computer Product Line Manager
Computer Sales Manager
Computer Applications Engineer.

1961
Instrument Associates
Arlington, Massachusetts

1957 to 1961
Sylvania Electric
Needham, Massachusetts

Design Engineer.

1955 to 1957
Hazeltine Electronics
Littleneck, Long Island, New York

Field Engineer

PROFESSIONAL SOCIETIES:

Member IEEE, AMA: Registered Professional Engineer (Massachusetts).

Richard J. Tringale

EDUCATIONAL BACKGROUND:

1966 to Present
Northeastern University Graduate School
Completed 75% of the course requirements toward a Master's Degree in engineering.

1961 Graduated
Northeastern University
Boston, Massachusetts
B.S. in E.E.

Graduated Malden Catholic High School.

EMPLOYMENT HISTORY

1961 to Present
DIGITAL EQUIPMENT CORPORATION
Position: Project Engineer

Currently employed at Digital Equipment Corporation in the Computer Special Systems group. Duties include design and project responsibility for interfaces on the 24, 250, 251, RM08 and RM09 drum systems; CDC 3200, 3300, and 3600 computer interfaces; original design work on the basic 680 communications system; and development of standard computer peripheral equipment including display and data communication systems.

1959 to 1961
HYPERION INCORPORATED
Watertown, Massachusetts

Testing and trouble-shooting of transistorized power supplies. These are regulared power supplies where only transistors are used. Drawing circuit diagrams and designing cable layouts and also modifying circuit designs. (Co-operative)

1957 to 1959
SANBOURN INSTRUMENT COMPANY
Waltham, Massachusetts

Repair and maintenance of electronic equipment used in recorders, such as electrocardiagraphs. (Co-operative).

Chapter 6
CCA Qualifications and Personnel

6.1 Computer Network Project

One of the earliest advocates of computer networks was J.C.R. Licklider, presently a Director of Computer Corporation of America. Dr. Licklider helped in the formulation of the concepts and in the promotion of the idea within the computer community*.

Early in 1965, Dr. Thomas Marill, President of CCA, proposed a feasibility study of computer networks, and a contract was awarded to CCA by M.I.T. Lincoln Laboratory, for such a study later in the year. A report** giving the results of the study, and recommendations for future action, was issued by CCA in 1966.

Some of the topics treated in this report were the following:

The concept of the computer network and its importance.
Available common carrier channels and interfaces, and comparative costs.
Software considerations, including software interfaces to existing time-sharing systems.
Recommendations for the implementation of an experimental network linking several computer installations across the country.

---

* Dr. Licklider became Director of Project MAC, M.I.T., in August 1968.

** A Cooperative Network of Time-Sharing Computers: Preliminary Study, Technical Report No. 11, Computer Corporation of America, Cambridge, Massachusetts (June 1, 1966).

As a result of this study, CCA proposed the implementation of an experimental network, and a contract was awarded to CCA by M.I.T., Lincoln Laboratory, in the summer of 1966. This experimental network was implemented and has been undergoing tests for some time. The details of the implementation and results of tests to date are contained in a CCA report*.

The network serves to interconnect three computers**: an AN/FSQ-32 running the SDC time-sharing system in California; the TX-2 computer running the APEX time-sharing system at M.I.T. Lincoln Laboratory; and a small, non-time-shared, display-oriented computer, the DEC 338, in the Pentagon. The link between TX-2 and the west coast is provided by the Western Union dial-up system (an automatic calling unit is used) at 1200 baud. Between TX-2 and the Pentagon, a 2400 baud line is leased from the telephone company.

As a result of three and a half years devoted to this project, CCA has accumulated a wealth of experience in areas such as the following:

Interfacing with telephone and Western Union facilities
Message formats
Coordination among remote sites
Automatic calling and answering units
Network check-out software

---

* An Experimental Computer Network, Technical Report No. 17, Computer Corporation of America, Cambridge, Massachusetts. (In preparation: Scheduled publication, November 1968). Prepublication copies (July 1, 1968) available from CCA.

** The interconnections are direct to the computers. There are no IMPs in the experimental network.

Maintenance of network software
Message protocol and error recovery
Remote debugging
Documentation of network software
Network demonstration programs
Software for measurement of network parameters
Code conversion software
Buffer management software

as well as the many small day-to-day problems that arise in dealing with computer networks.

6.2 Other Software Projects

CCA specializes in the design and implementation of advanced software. In addition to working on conventional data processing, CCA is currently working in the following areas:

Fingerprint recognition by computer
Computer languages
High-speed information retrieval
Deductive question-answering systems
Natural language processing

Software has been generated for a wide assortment of computers.

Two CCA proprietary software packages for information-retrieval applications have recently been introduced*: The Model 101 (for batch processing) and the Model 103 (for remote display terminals).

* See "Product of the Month" listing, Datamation February 1968, and Computerworld December 20, 1967, January 3, 1968, and August 14, 1968.

1. There are two protocols to be considered in the network: (1) the IMP-IMP packet protocol and (2) the IMP-HOST message protocol.

In general, protocol serves two functions: (1) The detection and correction of errors and (2) the control of buffers. The second function allows for quenching traffic when buffers are getting too full, and for stimulating traffic when buffers become free.

The first of these functions of protocol is required for IMP-IMP traffic but can probably be ignored in the IMP-HOST traffic. The second function is essential in both cases.

2. Consideration must be given to what the IMP receiving a garbled packet (that is, one having a bad cyclic check) is to do. Since the received packet is garbled, its identifying information is uninterpretable. Is the IMP to say only "Some packet I received from you (and it may have been an acknowledgement of one of my packets or simply noise on the line) was bad?" Such information would not be very helpful.

Here are some possibilities:

(i) Packets received correctly are acknowledged positively; packets received incorrectly are not acknowledged at all. Under such an approach, the transmitting IMP would retransmit his packet if no positive acknowledgement has been received after time t. The advantage of this approach is its simplicity. The disadvantage is that it is slow. With some negative acknowledgement protocol, the retransmission could be speeded up considerably.

(ii) Packets received correctly are acknowledged positively: packets received incorrectly are acknowledged with a packet that says "I have just received a garbled packet from you; the last good packet I received from you was such and such". In this approach, the transmitting IMP must keep a table of the identification of the last N packets it has transmitted to each of its neighboring IMPs. When one of these says "I have a bad packet from you; the last good one was such and such", the transmitting IMP can start retransmitting those messages which followed the last good one received. The advantage of this technique is that it, or some variant, can probably be made to work relatively fast. The disadvantage is its complexity.

3. What does an IMP do when it is running out of buffer space? (i) An IMP could tell all neighboring IMPs to wait before sending further packets, and later signal these IMPs when free buffers are again available. (ii) An IMP could choose to expunge certain packets already in its buffers, allowing the protocol later to retransmit these. (This course of action would require that the protocol guarantee that packets not be forgotten until accepted at their final destination.) (iii) Incoming packets could be ignored.

The second and third of these alternatives require retransmission of packets at a later time, increasing traffic loads. The first, however, requires signalling procedures which, by themselves, contribute to traffic.

4. At each moment of time, in regard to any packet p in the net, some IMP, say IMPi, must be responsible for seeing to it that p arrives at its assigned next destination. What happens

if IMPi discovers that it cannot fulfill its responsibility? (IMPi may discover, for example, that the only line assigned to the next destination for p has just gone out of commission.) Here are some possibilities:

(i) IMPi keeps trying to get p through. This is the simplest alternative. The disadvantage is that p may be jammed at IMPi for a long time, and that consequently the message is in transit (undelivered) for a long time.

(ii) IMPi simply gives up on p, and expunges this packet. The disadvantage is that the originating HOST will not know that its message has been lost.

(iii) IMPi sends a packet to the IMP associated with the HOST originating the message saying that the packet is undeliverable. The problem here is that it is not clear what the originating IMP is supposed to do at this point since it may no longer have the packet in its buffers.

(iv) IMPi sends a message to the originating HOST, saying that its message is undeliverable. This alternative requires (a) that IMPs be able to send messages to HOSTs other than their own and (b) that HOSTs be ready to accept such messages. If (a) and (b) are acceptable, this solution is probably the soundest.

2.5 Carrier-side Transmission and Reception

This section describes the queues and control information stored in module 0 of the IMP. It then describes operation of the module 0 software used to relay packets from one IMP to the next. Details of the operation concerning communication with the HOST-side software, routing, protocol, treatment and performance statistics are deferred to succeeding sections of this chapter.

Queues and Control Information

As mentioned in Section 2.2, packet buffers are contained in a common buffer pool in module 2. These buffers contain data only: control information for each buffer is held in a block of storage in module 0 called a Buffer Queue Element (BQE). The BQE contains:

1. A pointer to the next BQE in the queue in which the current BQE is contained.
2. The origin of the buffer itself.
3. The count of the number of words in the buffer which contain information rather than garbage.
4. The state of the buffer; this takes the form of the address of the routine which is next to operate on the packet contained in the buffer.
5. A word used for timing various operations, such as length of time allowed for acknowledgement of correct or incorrect transmission.

The BQEs for buffers that are not currently in use are chained to a two-word block called POOL: the two words point to the first and last BQEs in the buffer pool, respectively. The chain pointer (first word) of the last BQE in any queue contains zero.

The following is a list of control blocks (that is, blocks of consecutive words containing various types of control information) together with the queues that are attached to each type of control block. A brief statement of the purpose of each control block is included in the list. Note that all queues are composed of BQEs. (See Fig. 2.1.)

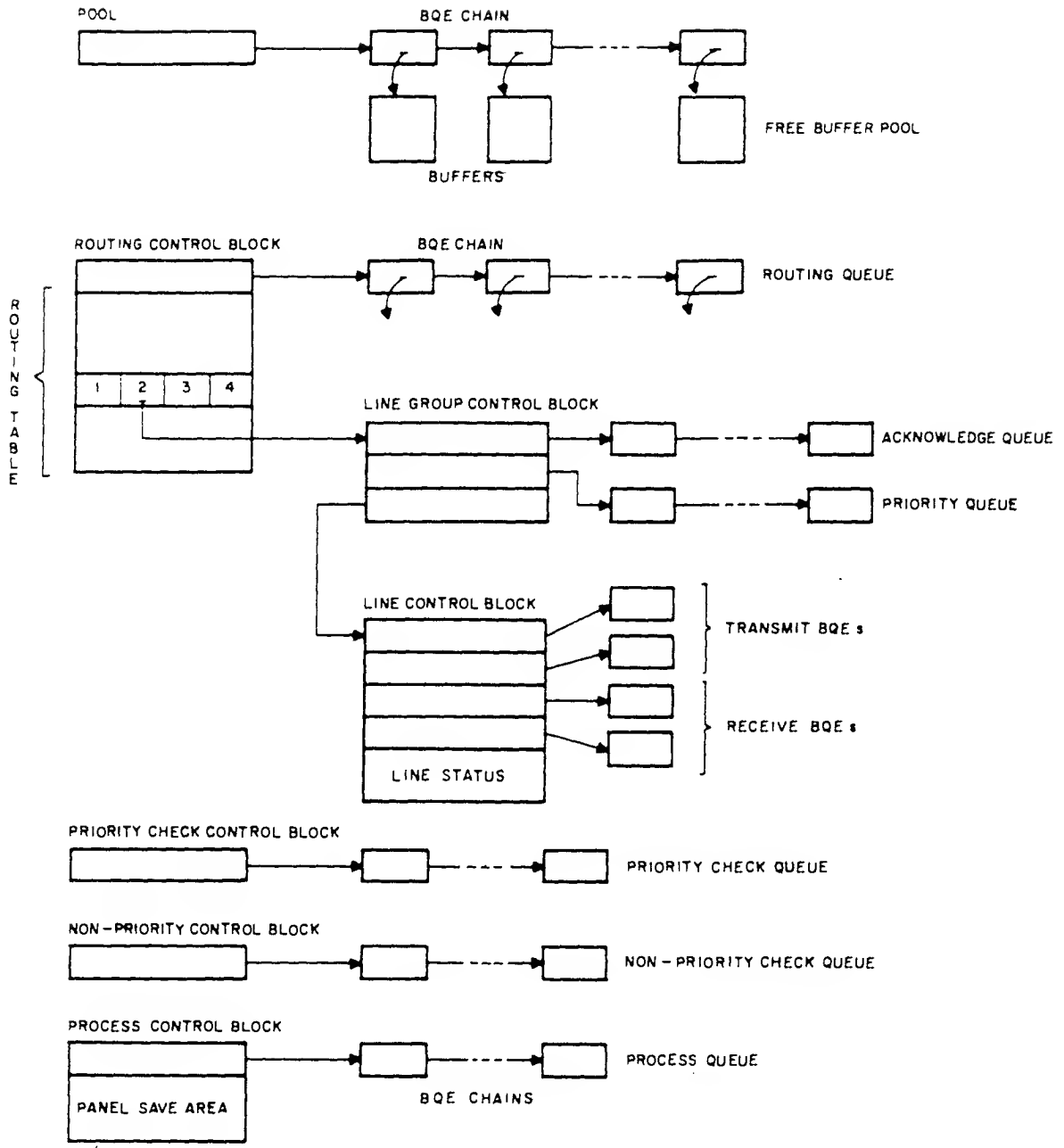


Figure 2.1 Control Blocks and Queues.

Buffer Pool Control Block (POOL): Used for

Free Buffer Queue: BQEs not currently in use.

Routing Control Block (RCB): Used to locate the Routing Table, and head of the

Routing Queue: BQEs for buffers which are to be transmitted but have not yet been attached to a LCB for transmission.

Line Group Control Block (LGCB): One for each group of lines going to the same IMP, used to locate the LCBs (see below) for the line group, and head of the

Acknowledge Queue: BQEs for buffers which have been transmitted over the line group but have not been acknowledged as correctly received.

Priority Queue: BQEs for buffers which contain priority packets for transmission.

Line Control Block (LCB): One for each line, used for storing line status information and pointers to the BQEs assigned for the current and next transmit and receive operations on the line.

Non-Priority Check Control Block (NCCB): Used for

Non-priority Check Queue: Low priority BQEs waiting for calculation of the 24-bit cyclic check sum.

Priority Check Control Block (PCCB): Used for

Priority Check Queue: Priority BQEs waiting for calculation of the 24-bit cyclic check sum.

Process Control Block (PCB): Used to save the CPU state during interrupt, and head of the

Process Queue: BQEs waiting for CPU processing.

Line information is contained in two types of control blocks. The Line Control Block (LCB) exists for each full-duplex line, while there is one Line Group Control Block (LGCB) for each group of lines having the same destination IMP. Each LCB contains:

1. The address of the first hardware location (the mailbox) used to control transmission of words between the line unit and storage (see Section 1.4).

2. A word containing the line address and flags used to set status of the line unit and modem.

3. A word into which current status of the line unit and modem is read.

4. A pointer to the LGCB for this line group.

5. Pointers to the two (maximum) BQEs selected for reception over the line. Normally, the first is in the process of being received and the second is next in line for receiving. The corresponding buffer origins and word counts are posted in the receive mailbox for the line.

The LGCB contains:

1. Pointer to the first and last BQE corresponding to buffers which have been transmitted but whose correct receipt has not been acknowledged*.

---

* BQEs are normally placed at the tail of a queue and taken from the head of the queue.

2. Pointer to the first LCB for this line group and count of the number of LCBs in the group.

3. Pointer to the first and last BQE corresponding to buffers containing high priority packets for transmission.

The routing table contains a four-word entry per node in the network. Each word contains a zero if no route exists and otherwise contains a pointer to the LGCB for the corresponding route. During routing, the four words are tested in order to find an LGCB which can be used for transmission. If none is found, the BQE will be placed on the queue chained to the Routing Control Block (RCB). This contains:

1. The origin and length of the routing table.
2. Pointers to the first and last BQEs in the routing queue.

As BQEs are removed from transmit queues, the priority queue and the routing queue are tested to find BQEs that can be assigned to a LCB for transmission.

The above control blocks and queues contain the basic information necessary for control of routing, transmission, and reception. Three more queues (of BQEs) exist: the Process Control Block (PCB) is used to queue up BQEs which require CPU processing before any further action can take place, while the Priority Check Control Block (PCCB) and Non-Priority Check Control Block (NCCB) are used to queue up BQEs which are waiting for use of the cyclic checking hardware. Each of these control blocks contains a pointer to the first and last BQE in its queue. In addition, the PCB contains words used to save the contents of the accumulator, link register, instruction field register, and data field register during an interrupt.

When the CPU is not processing an interrupt, it is executing the routine pointed to by the BQE which is at the top of the process queue. After execution of that routine, return is made to the CPU dispatching routine indicating:

1. Which queue the BQE is to be placed upon.
2. Which routine is to be executed the next time the BQE appears in the process queue.
3. Neither action is required.

The third case may occur since interrupt routines and other module 0 routines alter the process queue; the module 1 routines, however, may not do this.

Store and Forward Operation

This section describes the chain of events including reception of a packet, checking, header analysis, routing, transmission to the next IMP, and receipt of acknowledgement. Each event which initiates processing by means of an interrupt or by placement of a BQE on the process queue is described, followed by a description of the subsequent processing. Abnormal processing (such as receipt of a NAK message) is not described in this section.

Carrier On: Detection of the carrier by the modem indicates that the far end transmitter is in operation. This causes the line unit multiplexer to request an interrupt. The IMP, in order to service the interrupt, issues an IOT instruction which loads the accumulator with the line unit status word, containing the line number and status flags. Using the line number, the IMP locates the proper LCB and, by comparison of the new status information with the old status information, discovers that Carrier On

was the cause of the interrupt. It takes two BQEs from the pool, posts their origins and maximum word counts into the receive mailbox for the line, places pointers to the BQEs in the LCB, and issues an IOT with a set status word for the line which starts reception. The automatic buffer switching at the end of the first packet is also enabled. After establishing line synchronization, the line unit multiplexer will place the first packet into the first buffer, request an interrupt when transmission of the first packet has completed and it has switched the line unit to read into the second buffer, and disable automatic buffer switching.

Buffer Loaded from Line: The IMP reads the line status, locates the proper LCB and BQE, and sets the actual word count (using the contents of the receive mailbox) into the BQE. At this point, the entire packet (including the 24-bit cyclic check sum) is in the buffer, but the check sum has not been checked. In order to do this, the IMP places the BQE in the high or low priority (as determined from the header) check queue using the PCCB or NCCB. If both the check queues were empty, the IMP initializes the cyclic checker receive mailbox and issues the IOT which starts the checking process for receive. The IMP now obtains a BQE from the buffer pool, places a pointer to it in the LCB, initializes the receive mailbox, and issues the set status IOT in order to enable the automatic buffer switching.

Cyclic Checking Complete (Receive): The IMP issues an IOT to test the result of checking. If the check is successful, the BQE is posted with the name of the Acknowledgement Routine. If the check was unsuccessful, the packet header and the word count in the BQE are altered appropriately (including setting priority bit) and the BQE is posted with the name of the Transmit Check

Start Routine. In either case, the BQE is placed in the process queue. If either check queue is not empty, the IMP now starts checking for the next queued BQE.

Acknowledgement Routine: The BQE is at the head of the process queue and represents a correctly received packet. The IMP first obtains a BQE from the buffer pool and writes an acknowledgement message packet into it. This BQE is posted with the name of the Transmit Check Start Routine and placed in the process queue following the first BQE. The Header Analysis Routine is now entered using the BQE at the head of the process queue.

Buffer Emptied onto Line: This event is signalled by an interrupt. The IMP reads the line status and locates the proper LCB and BQE. The header is checked to see if acknowledgement is expected. If not, the BQE is restored to the buffer pool, and otherwise it is placed on the acknowledgement queue for the line group, using the LGCB pointer in the LCB. The priority queue attached to the LGCB is now checked to see if it has any BQEs on it. If so, the first BQE is chosen to be assigned to the LCB. If not, the routing queue is scanned to find the first BQE which can use this line group for transmission, using the routing table. If, by either method, a BQE is found which can use the line, it is chained to the LCB, the mailbox is initialized, and the set status IOT is used to start transmission or to enable the automatic buffer switching at the end of the current transmission.

Transmit Check Start Routine: The BQE at the head of the process queue points to a buffer containing a packet which was altered or is originated by this IMP. The IMP places this in the appropriate check queue. If both check queues were empty, the

IMP issues the IOT which starts the checking process for transmit. The Transmit Check Start Routine now exits to the CPU Dispatching Routine to start processing the next BQE in the process queue.

Cyclic Checking Complete (Transmit): The 24-bit check sum has now been calculated and placed in the proper position in the buffer. The BQE is now placed on the process queue after being posted with the name of the Routing Routine, and the checked queues are interrogated to see if another buffer is waiting for the cyclic check hardware. If so, the IOT is issued to start the next check sum calculation.

Header Analysis Routine: The IMP first checks to see if the packet is destined for a local HOST. If so, the HOST Packet Reception Routine is called (see Section 2.6 for description of further action). If the packet was destined for this IMP, the IMP Packet Reception Routine is called (see Section 2.8). Otherwise, the Routing Routine is entered to forward the packet to the next IMP in line.

Routing Routine: Using the BQE at the head of the process queue, the IMP obtains the destination code from the buffer and enters the routing table. If the priority bit in the header is set, the most direct routine for which a line is in service is chosen, and the BQE is attached to the priority queue for the corresponding LGCB. Otherwise, the BQE is attached to the end of the routing queue. In either case, an attempt is made to complete routing and find a line that can be used immediately. If one is found, rather than attach the BQE to the routing queue or priority queue, the Transmit Routine is entered to attach the BQE to the LCB, initialize the transmit mailbox, and issue the

set status IOT in order to start transmission or to enable automatic buffer switching at the end of the current transmission. The routing routine now exits to the CPU dispatching routine to process the next BQE in the process queue.

Acknowledgement Packet Received: This event is recognized by the IMP Packet Reception Routine. The problem now is to find the BQE in the acknowledgement queue corresponding to the acknowledgement packet. This can be done by scanning buffers and matching the header information against that of the acknowledgement packet. This will work only if the origin code of the acknowledgement packet is that of the original origin of the packet being acknowledged, rather than that of the IMP originating the acknowledgement, but it is fairly slow. Instead, it is proposed that the "handover number" field of the ARPA specifications be used to record an encoded form of a pointer to the BQE for the packet whose receipt is to be acknowledged. (The handover number is not necessary for use in routing control, if the routing method proposed is employed.) This BQE is removed from the acknowledgement queue and restored to POOL together with the BQE for the acknowledgement packet. Exit is then made to the CPU Dispatching Routine.

2.6 HOST-side Transmission and Reception

As seen in Section 2.5, the HOST-side software acquires packets from the carrier-side software by placement of their BQEs in the process queue and subsequent entry into the HOST Packet Reception Routine. Conversely, the HOST-side software causes a packet to be transmitted by exit to the CPU Dispatching Routine with the BQE posted with the name of the Transmit Check Start Routine.

HOST-side software may also execute certain routines in the carrier-side software directly, such as the routines which move a BQE between the buffer pool and other queues.

Initiation of transmission and reception over the HOST-IMP line(s) will normally occur at interrupt time; it is desirable that this part of the HOST-side software be protected from the effects of bugs in other portions of the HOST-side software, which is expected to be subject to more frequent change and extension. In addition, it will be necessary to move BQEs between queues at interrupt time, and highly desirable that this be done only by module 0 routines. For these reasons, it is proposed that the HOST-side software concerned with issuing IOTs and interrupt-time actions be housed in module 0 rather than module 1.

The standard HOST-side software supplied will assume that the option 1 method of attachment (as described in Section 1.7) is used. With obvious exceptions, such as the absence of cyclic check sums, the organization of software controlling transmission over the HOST-IMP interface will be similar to that described in Section 2.5. The principal difference is related to the necessity for blocking and unblocking messages. This subject is discussed below.

Standard packet buffers in module 2 are used for transmission and reception over the interface, using the automatic buffer switching feature of the line unit multiplexer. The message header and TEXTBLOCK 1 (cf. Section 2.4) are received in the first packet buffer, while the receive mailboxes for subsequent TEXTBLOCKs are initialized to cause transmission into the TEXT portions of the packet buffers, leaving the headers to be filled in by a routine

that is executed from the process queue rather than at interrupt time. At this time, any TEXT conversion and reformatting required is done as well. The converse type of operation occurs for transmission of messages from the IMP to the HOST.

The complication during reception of messages for the HOST occurs due to the fact that packets may be received out of order. This is taken care of by queuing up packets on the HOST-side in packet number order as they are received from the carrier-side software, and initiating transmission to the HOST only when all packets for a given message have been received. This could still conceivably result in messages being received out of order! This situation can be detected, and often corrected, if the convention that Message IDs from a given origin are monotone increasing with time is observed.

In the case that multiple HOSTs are connected to the IMP, the method of picking the appropriate HOST-side software for handling HOST-IMP interface transmission and processing routines is quite straightforward. Consider, for instance, the action of the carrier-side Header Analysis Routine: it determines that a packet terminates at this IMP on the basis of matching the destination code against a table containing its code and that of its HOSTs. The table also contains the address of the proper routine to call to handle the packet. Similarly, on reception of a packet buffer load from the HOST, the HOST-side software (at interrupt time) can pick up from the HOST LCB the name of the HOST routine to be posted into the BQE as it is placed into the process queue.

2.7 Routing and Traffic Control

The nature of the way in which the carrier-side software handles routing has already been indicated in Section 2.5. Briefly, a routing table is used which has an entry per destination HOST and/or IMP. The entry contains pointers to up to four line groups (note that we have assumed that there may be enough traffic to justify more than one line between some pairs of IMPs), in order of preference. Nothing has been said about the way in which the routing table is formed: this subject is raised in Section 3.5.

Priority traffic is always queued up using the line group with the highest preference for the given destination (priority packets are used only for system purposes, such as transmittal of acknowledgement packets). Non-priority traffic may end up using lower preference routes, depending on current line usage and the order in which lines become available for dispatching additional packets.

It should be noted that two choices of routing algorithm, assuming the above organization of the routing table, are possible. The problem can be illustrated in the following way: Suppose that a line becomes available for dispatching (that is, the current buffer has just finished being transmitted and the next buffer transmission is started by the automatic buffer switching feature of the line unit multiplexer). Suppose further that the only two BQEs on the routing queue that can use the line at all are the first and third BQE, but the first BQE has second preference for the line while the third BQE has first preference for the line. Which BQE should be dispatched? The simplest method to implement is the one indicated in Section 2.5--dispatch the first BQE that can use the line at all. The effect of this choice will be that

some packets will be sent over alternate routes, whereas they could have been sent over more direct routes if they had waited for one to become available. On the other hand, the order of dispatching will be closer to FIFO with this choice. Another consideration will appear below.

When a line goes down, what should happen? We believe that recalculation of a new Routing Table on the spot is infeasible; this is a fairly major operation, and should be done using as complete information on network traffic as is possible. Routing Tables should also be calculated for the network as a whole, not just for one IMP at a time. We have therefore assumed that the Routing Table will remain unchanged over reasonably long periods of time. This being the case, the effect of a line going down is made to appear as if the line is constantly busy, but no change in the routing algorithm is made--the existing alternate routes are used. The difference between a line being down and its being constantly busy is important for only one purpose. Namely, the choice of line group for priority traffic will be the highest preference line group containing lines which are up.

Having said the above, another reason for the choice made further above appears: if we had chosen the BQE with highest preference for the line group rather than the BQE nearest the head of the routing queue, traffic which had first preference for a line group that is down would tend to get crowded out in competition for lower preference line groups.

2.8 Control Messages and Packets

Control messages are those messages destined for or originated by an IMP. (Those destined for IMPs are handled by the IMP Packet Reception Routine, mentioned in Section 2.5). This section lists possible message types, with brief indications of their purposes:

1. Acknowledge: Acknowledgement of packet received by IMP, sent to transmitting IMP as a high priority message.

2. Negative Acknowledge: Statement by IMP to the transmitting IMP that a packet has arrived which is invalid (normally because the check sum is incorrect), sent as a high priority message.

3. Enquiry: Sent by IMP to a neighboring IMP, as a high priority message, as the result of a failure to receive an acknowledgement, a status report, or a Resume message within certain time limits.

4. Hold: Sent by IMP to a neighboring IMP as a high priority message. The purpose is to reduce a peak IMP traffic load or to avoid receiving more packets following one that cannot be disposed of (e.g., no available forwarding route due to line outages) for a period of time.

5. Resume: Sent by IMP to a neighboring IMP as a high priority message to cancel the effect of a previous hold.

6. Send Status: Causes receiving IMP to report the current status of its hardware and software. Contains items such as which lines are up and down, what statistics are being gathered (see Set Status, below), etc.

7. Here Is Status: Contains IMP status information requested by Send Status message.

8. Set Status: Causes receiving IMP to set and/or reset certain status flags. E.g., put line up or down, start or stop collecting certain statistics (probably selectively), etc.

9. Send Statistics: Forward IMP-collected statistics to originator of this message.

10. Here Are Statistics: Contains the current values of statistics being gathered by the originating IMP.

In addition, the ability to load certain system data, such as a new routing table or IMP routine, will be provided. The form in which the data is loaded as well as the restrictions which must be placed on such data will be determined during the system design process.

2.9 Measuring Network Performance

The IMP must collect statistics which are used for basically three purposes:

1. Detection of equipment malfunctions.
2. Detection of IMP overload or near-overload.
3. Engineering the network configuration and calculation of routing tables.

The basic methods of collecting statistics and detection of malfunction or overload will be by accumulation of counts--either totals or counts of a certain type of event within a given time period. For instance, if the average packet length is known, an estimate of line load can be obtained by counting the number of packets transmitted within a given period of time. One could,

for instance, set the expected number, plus some margin, into a counter each 1/60th of a second and conclude that a near-overload has occurred if the counter goes through zero before the counter has been reinitialized. A more refined method of estimation would be to accumulate total number of words transmitted in a given time period. Another method of estimation would be to accumulate the total amount of time a facility is idle by incrementing a counter at each clock tick if the facility is idle. Similarly, timers will be used to signal a network or equipment malfunction. For instance, non-receipt of an acknowledgement message within a given time (which may depend on the particular line group in use) signals the existence of a network overload or malfunction and the need for corrective action.

For engineering purposes, however, a knowledge of distributions rather than averages may become important. Due to the limited storage capacity of the IMP, information recorded on a per event basis (such as the distribution of packet lengths) will be collected in a statistics buffer for later transmission to a selected HOST on buffer overflow. Other statistics that might be handled in this way would be items like a periodic recording of queue lengths.

An especially important quantity to keep track of is the number of buffers in the buffer pool. This must be kept from completely emptying out, since it will be necessary to obtain one or two buffers to send control messages and receive responses to them during periods of overload. The system must be designed so as to fail safe, rather than collapse, at overload!

Among the statistics to be accumulated, either unconditionally or optionally (under control of the Set Status control message), are:

1. Count of errors in transmission, per line per direction.
2. Same, per some time period, for detection of equipment requiring repair.
3. Count of packets transmitted per line per direction.
4. Same, per time period, for load recording and overload control.
5. Buffer pool size.
6. CPU load: HOST-side, carrier-side, and interrupt processing.
7. Timing for all actions requiring responses, and count of number of timeouts.
8. Message and packet lengths.
9. Transmission acknowledgement delays, as a measure of total message delay.
10. Queue lengths.
11. Number of packets sent by routes of different preference, per destination.

2.10 Fault Recovery

In earlier sections, the types of mechanisms employed to detect faults have been briefly discussed. The principal sources of fault are:

1. Transient transmission errors introduced in the common carrier lines and equipment.

2. Permanent common carrier line and equipment failures.

3. IMP hardware failures.

4. IMP software bugs.

5. Failure to receive packets correctly due to system overload resulting from contention for memory cycles, interrupt processing, or free buffers.

Items 1 and 5 above are cured by retransmission of packets, while the remaining items require manual intervention for cure. The frequency of faults due to items 3 and 4 is expected to be unusually low due, on the one hand, to the unusually long mean time to failure of the DEC PDP-8I and, on the other hand, to the modular design of the CCA IMP software.

The effect of transmission errors exceeding a certain level during a given time period will be to cause the IMPs at either end of the line to shut it down and call for manual intervention. If an IMP itself gets into trouble, the effect will be that neighboring IMPs will observe invalid packets and/or excessing delays for packet transmission acknowledgement. In this case, the neighboring IMPs may be uncertain whether the condition is due to the IMP at the other end or to the equipment. In either case, however, manual intervention must be requested by the IMPs observing the abnormality. Flow of information in the network is maintained, where possible, due to the use of alternate routing of packets.

Because of the long mean time to failure of the DEC hardware, together with the availability of the power failure and automatic recovery options, it is feasible to run the IMP unattended, with power on at all times. Whenever power goes down at an IMP, it will be detected by neighboring IMPs due to the disappearance of

the Carrier on flag at the line units. Again, a neighboring IMP may be uncertain whether this is due to IMP or line failure, but rerouting of traffic and a request for manual intervention may be done automatically in either case.

The handling of a request for manual intervention may be done as HOST-side option. The major possibilities are:

1. Print a message on the attached IMP teletypewriter.
2. Relay a message to the local HOST.

In addition, it is proposed that either action be accompanied by sending a Here Is Status control message to the network HOST computer acting as the sync for statistics.

Chapter 3
Network Performance Analysis

3.1 Introduction and Summary

In this chapter, data in the ARPA specifications, together with parameters estimated for the proposed system, are used to calculate network performance. The calculations are based on a queuing theory model which is expected to yield conservative results for delays and load handled.

Data for the model contained in the ARPA specifications, section F(4), is as follows: A symmetric, with respect to nodes, network is postulated. The numeric quantities are:

$m = 3$	Number of links traversed by message
$k = 4$	Number of links at each node
$I = 20KB$	HOST-IMP input rate
$L = 15KB$	Link transmission rate (one way)
$N = 364$	Number of bits per packet
$L_L = 300$ mi.	Link length
$D_L = 5.5$ μs/mi	Delay per mile
$D_M = 760$ μs	Modem delay
$D_{CL} = 3.17$ ms.	Communications delay per link
$D_T = 7.28$ ms.	Transmission delay per link

The significant estimated parameters for the proposed IMP are:

$T_P = 1$ ms.	CPU processing time per packet
$T_M = 496$ μs.	Memory usage per packet transmission

The latter quantity represents the memory time used during packet receipt, transmission, and cyclic checking; this is unavailable for use by the CPU for processing.

The analysis provides the following significant resulting quantities: The resulting maximum CPU capacity is

$$C_C = 676 \text{ packets/sec.}$$

The average packet delay for the ARPA model between originating and terminating HOST, including the above delays and queuing delays is

$$D = 67.1 \text{ ms.}$$

The probability of exceeding a total packet delay of 0.5 sec. is 0.0035, approximately.

The maximum HOST-IMP input rate with all other nodes quiet is calculated from C_C, above. This is 2420 KB. It should be noted, however, that loading the CPU to full capacity would result in unacceptable queuing delays.

3.2 Queuing Model

The queuing model used is M/M/c:(∞, SIRO). That is, the input traffic interrival and service time distributions are assumed to be exponential, with infinite queue capacity and service in random order by c servers. The calculations use curves for this model which are based on Riordan's work and are published in

Wilkinson, R.I., "Working Curves for Delayed Exponential Calls Served in Random Order", BSTJ 32 (1953), pp. 360-383.

The principal quantities appearing in the theory are:

λ = number of demands for service per second
h = average service time per demand (holding time)
c = number of servers
α = proportion of time a server is busy = $\lambda h/c$
$\overline{t}$ = average delay before service
$P(>0)$ = probability that a demand for service is delayed
$P(>t)$ = probability of delay longer than time t.

Based on the network parameters presented above, the input load in packets/sec. for various numbers of links per node is:

k	λ
1	209.4
2	299.2
3	389.0
4	478.8
5	568.6
6	658.4

3.3 CPU Queuing

Of the various queuing delays encountered, the CPU queuing delay is the most critical, given sufficient links per IMP to handle the traffic. The CPU is used a number of times in handling a packet: there are two interrupt services per packet transmitted

or received (due to the use of the proposed method of accomplishing the cyclic check) as well as the time required for packet processing between receipt and retransmission of a packet. The results for assumed total processing time of one millisecond are presented below:

k	λ	h	α	$\bar{t}$
4	479	1	0.48	0.9

The probability of exceeding 44 ms. delay is 0.001.

3.4 Buffer Queuing

As stated in Chapter 2, a pool buffering system is used. Depending on the number of buffers provided, packets may be held or retransmitted due to the unavailability of input buffers. To determine the probability of this occurring (which, by the way, is independent of the service time distribution, provided that the demands occur at random times), we must find $P(>0)$, given the expected buffer holding time.

A buffer is in use from the time the input packet starts being received until it is retransmitted and acknowledgement is received from the HOST or IMP to which it is forwarded. In the latter case, we encounter three transmission delays and two communication delays. Assuming two milliseconds for packet processing and two milliseconds for interrupt processing (a total of seven interrupts are significant for this part of the CPU delay for the two IMPs concerned), the total buffer holding time is 32 ms. However, the average CPU holding time should include the expected CPU delays for this calculation. The significant delay is 0.9

ms. for the CPU packet processing. Thus, the average buffer holding time is 39.4 ms. With 31 buffers, $P(>0)$ is 0.018 and the percent of time each buffer is busy is $\alpha = 0.61$.

3.5 Routing Doctrine

The method of routing proposed is that used in the Bell System for toll traffic. Briefly, this doctrine establishes a hierarchy of switching points which provides a final alternate route by ascending the hierarchy to the "national center" from the originating point and then descending the hierarchy to the terminating point. Links are provided between nodes in the network in numbers justified by the predicted direct plus alternate-routed traffic between each pair of nodes, for all end-to-end routes using that subpath. Routing tables are then prepared for all nodes. These tables use the most direct route toward the terminating node for the first choice, the next most direct for the first alternate route, etc.

The theory underlying this doctrine is discussed in

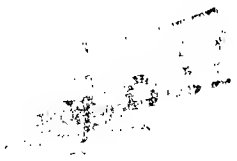
> Wilkinson, R.I., "Theories for Toll Traffic Engineering in the U.S.A.", BSTJ 35 (1956), pp. 421-514,

and methods are given for determining the network configuration.

#9

DIGITAL EQUIPMENT CORPORATION

INTERFACE MESSAGE PROCESSORS FOR THE ARPA COMPUTER NETWORK



September 5, 1968

Technical Proposal Submitted to:

Department of the Army
Defense Supply Service--Washington
Room 1D 245, The Pentagon
Washington, D.C. 20310

Attention: Mr. Daniel B. Dawkins

Request No.: DAHC15-69-Q-0002

DIGITAL EQUIPMENT CORPORATION

MAYNARD, MASSACHUSETTS

KENNETH H. OLSEN
PRESIDENT

August 29, 1968

Department of the Army
Defense Supply Service--Washington
Room 1D 245, The Pentagon
Washington, D. C. 20310

Attention: Mr. Daniel B. Dawkins

Gentlemen:

Digital Equipment Corporation takes pleasure in submitting its proposal for the Interface Message Processors for the ARPA Computer Network (Request No. DAHC-15-69-Q-0002). The formation of the computer network planned by ARPA is of great interest to our Company, and to me personally, and I would like to express my hope that DEC will have the opportunity of working on this forward-looking project with ARPA.

Sincerely yours,

Kenneth H. Olsen

KHO:ecc

Preface

A proposal by Digital Equipment Corporation for the Interface Message Processors for the ARPA Computer Network is presented. The proposed hardware and software is discussed in detail, and an analysis of network performance using this hardware/software combination is presented.

The qualifications of Digital Equipment Corporation for the present task are discussed. DEC's history of accomplishments in the small computer field is reviewed, with emphasis on the company's extensive experience in the data communications field and in the implementation of special systems configured to customer requirements.

Software will be provided by Computer Corporation of America under subcontract. This company was selected as being the best qualified for the project, because of its unique experience with computer networks, its thorough acquaintance with DEC equipment, and its demonstrated capability in implementing imaginative, advanced software.

Table of Contents

Table of Contents (Continued)

List of Illustrations

Chapter 1
Hardware

1.1 Introduction

The hardware proposed for the IMP is shown in the attached block diagram (Fig. 1.1). A discussion of each logical element of the system is given in succeeding sections of the present chapter. It will be noticed that three alternative options for the HOST-IMP interface are presented. In the remainder of this proposal we assume that option 1 has been selected. If option 2 or 3 is selected at a particular installation, DEC will be prepared to work with that installation to engineer a suitable interface. Such special work, however, does not form a part of the present proposal.

1.2 Central Processor Unit

The central processor proposed is a PDP-8I with the addition of several options.

PDP-8I

The PDP-8I is a small scale general purpose computer. It is a one-address, fixed word-length, parallel computer, using 12 bit two's complement arithmetic. Cycle time is 1.5 microseconds. Core memory of up to 32,768 words can be accommodated.

Power Fail Option Type KR01

The power fail option is physically implemented within the PDP-8I processor. This option indicates to the program when the A.C. line voltage has gone below an acceptable threshold. Upon detection of this condition, the program is able to store all active registers and system status in core memory so as to prevent loss of data.

When the AC line voltage returns above the acceptable threshold, the computer is automatically started at a fixed location, with no operator intervention required. This option allows the IMP to tell adjacent IMPs that it is disconnecting from the network, and, at a later time, to indicate that network participation is being resumed.

Automatic Recovery Option Type KR08

The automatic recovery option type KR08 was designed for environments which require unattended system restart of a PDP-8. The most common application is in 680 systems which are used as remote line and message concentrators for time-sharing systems.

The operation of the KR08 is based on two delays which can be manually adjusted to clock out after a desired time interval which begins at the time the KR08 is pulsed. The first delay is set to a short period of time and the second delay is set to a long period of time. The program issues a special instruction at regular intervals. This instruction provides "initiate pulses".

If, due to a hardware or software failure, the interval between initiate pulses becomes less than the short delay or greater than the long delay, the KR08 will stop the processor, place 18 fixed

recovery instructions into preselected memory locations and restart the PDP-8I at a preselected address. This option can be used to protect against inadvertent loops and halts in the HOST-side software.

The KR08 automatic recovery option can also be initialized by push-button action or by an external level, such as the ring signal for a modem. A manual switch is provided to inhibit KR08 action.

I/O Bus

The PDP-8I's I/O bus is designed so that all device selection can be done within the I/O device, requiring no modification to the processor when a new I/O device is added.

Device selection is achieved by providing a synchronizing pulse, at which time the device decodes the contents of address lines to determine device selected. The contents of the address lines are under program control, thus achieving full program control of device selection. The logic necessary to detect device selection is available from DEC as a single module.

Data transfers between the I/O devices and PDP-8I memory can be accomplished in one of three modes.

(1) I/O Transfer. In this mode of operation data is transferred by I/O instructions within the program. Data to the device is read from 12 accumulator status lines, which are made available to the device via the I/O bus. (The program loads the accumulator with the data to be transferred to the I/O device.) Data from the device is read into the PDP-8I accumulator when the I/O read command

is issued by the program and decoded by the device. All gating into the accumulator is under control of the I/O device, thus permitting maximum flexibility.

(2) Three-cycle Break. The three-cycle break mode of operation provides an inexpensive means by which the I/O device can transfer data in and out of the PDP-8I's memory, without the need for PDP-8I program control. However, in most applications, data transfers and block lengths are controlled and initiated under program control.

When an I/O device desires to transfer data in and out of PDP-8I core memory, it places a transfer request level on the bus along with a level which indicates direction. When synchronization is established with the PDP-8I memory, a fixed location (hard wired within the device) in PDP-8I memory is fetched, the contents are incremented by one and, if the result is zero, a "word count = 0" pulse is returned to the I/O device. During the next memory cycle the next consecutive location is fetched, its contents incremented, and the result used as the absolute address from which a 12-bit word is transferred in or out of the PDP-8I's memory.

(3) Single-cycle Break. The single-cycle break can be considered identical to the three-cycle break, except that the word count and transfer location are maintained by the I/O device, thus requiring only one memory cycle per transfer instead of three.

Real Time Clock

A real-time clock will be provided, consisting of a 12-bit counter which is loaded with the 2's complement of a number N. The counter is incremented every millisecond. When overflow occurs at N milliseconds, a program interrupt occurs. The clock can be enabled and disabled under program control.

Extended Arithmetic Element Type KE8/I

This option consists of circuits that perform parallel arithmetic operations on positive binary numbers. A 12-bit multiplier quotient register (MQ), a 5-stage step counter (SC), and various shifting and control logic constitute the option. The presence of this option facilitates the many shifting operations necessary in the IMP application.

1.3 Data Channel Multiplexer

The data channel multiplexer type DM01 provides multiplexing of up to 7 devices for access to PDP-8I memory via three-cycle or single-cycle data break. Multiplexing is on a priority basis with priorities being established physically. In the proposed system the multiplexer is used to service the line unit multiplexer on a three-cycle break basis and the cyclic check calculator on a single-cycle basis.

1.4 Line Unit Multiplexer

The line unit multiplexer (type DC08S) is used to multiplex up to 10 full-duplex synchronous line units into the PDP-8I via the three-cycle data break feature. Each line has two data transfer units: one for sending data and the other for receiving data. Using the three-cycle data break a block of data is transferred by providing word count (WC) and current address (CA) location in memory, pointed to by the external device. Each data transfer unit operates independently and, to allow for block chaining, can point to two WC/CA's or "mailboxes" in memory. On completion of the transfer of one block of data, a new block can be started automatically by the line unit specifying the other WC/CA. With this feature the operating program has a full block-transfer time to specify a new WC/CA for a new block of data (and to set a control bit in the line unit to indicate that automatic transfer to the new WC/CA is desired at the completion of the current block transfer).

The multiplexer has a free-running scanner to gate break-requests into the 8/I. It also has the ability to queue up interrupts and control signals. When a line unit has a character to be placed in memory, a receive break request is gated onto the PDP-8I break request line by the free-running scanner. This request will be answered by the PDP-8I and the word inserted into memory. When the line unit needs a character to be transmitted, a transmit break request is gated onto the PDP-8I break request line by the free-running scanner. This request will be answered and a character will be removed from memory and transmitted. Each time a break request is generated, one of the prewired addresses associated with that line is gated into the memory address register. At this location is located an absolute address from which or to which a character will be removed from memory or placed into memory.

The multiplexer also controls the order of program interrupts (PI). These are gated onto the the PDP-8I bus by the free-running scan check. When a PI is gated onto the bus, the master status register is filled with the status of the line unit causing the PI. At the time the status is gated into the master status register, the initial cause of the interrupt is cleared, and succeeding interrupts will be noted by the multiplexer but not acknowledged until the first interrupt is serviced. Servicing of the program interrupt takes place when the IOT SKIP ON MULTIPLEXER is issued. When this IOT is issued, the contents of the status register will be gated into the AC and the program will skip the next instruction. The status register is constructed so that the octal address will appear in 4 bits of the AC and other bits will contain the following information.

TRANSMIT STATUS: If a 1, the transmit logic is actively transmitting data. If a 0, the logic will be idling and either the logic or the modem will be supplying a SYNC character to the transmit data lead.

RECEIVE STATUS: If a 1, the receive logic is actively receiving data. If a 0, the logic will be seeing SYNC characters and maintaining sync.

CONTROL CHAR: If a 1, the logic has detected a special character combination. If transmitting, the characters were DLE ETX. If receiving, the characters could have been either DLE STX or DLE ETX. It will be up to the software to know which one.

REC/TRANS: If a one bit, the logic that caused the control char interrupt was on the receive side of the line. If a zero, the interrupt was caused by the transmit side of the line.

CARRIER ON/OFF: If a one bit, the carrier has gone off. If a zero, the carrier is on.

CLR TO SEND: If a 1, the transmit logic is Clr to Send. If a 0, the transmit logic is not Clr to Send.

INTERLOCK: If a 1, the interlock is in effect. If a 0, the interlock is not on and the modems are not completely in sync.

MBX ACTIVE: Indicates which of the two WC/CA's ("mailboxes") is currently active.

The multiplexer also contains a master control register. This register is loaded from the AC of the PDP-8I when the IOT Load Control Register is issued. This register will be loaded with the octal address of a line unit whose control register is being changed.

The control bits will perform the following functions:

IDLE: If set along with the transmit bit, the send active flip-flop will be cleared and the request to send will be removed from the modem. When this request is removed, the modem will send sync characters on the transmit data lead.

If set along with the receive bit, the Rec active flip-flop will be cleared and the receive logic will not receive any more text until the beginning of the next message.

DISABLE THE LINE: If set, either the send or receive side of the line will be prevented from requesting a break from the PDP-8I. (Normally used because a fault condition is present.)

If reset, the side of the line will be enabled.

TRANS/REC: If set, the receive logic will be the side of the line affected by the control register.

If reset, the transmit will be the side affected.

CHANGE MODE: If set, either the receive or the send side of the line will not delete or add "DLE" characters depending on the side affected.

PRIME ENABLE: If set, the feature to allow a second buffer to be transmitted or received immediately following another buffer will be enabled.

If a general check of the system is required, the status of each line can be found by issuing the Read Status Register IOT. To do this the octal line number is loaded into the AC and the IOT issued. The AC will be loaded with the status bits for the line.

1.5 Line Units

The line units (type DC08T) receive the serial data from the Rec data lead of the modem and place serial data onto the Xmit data lead of the modem. Each line unit consists of three interrelated sections.

I. The Control Section

The control section has the task of coordinating the activities of the modem functions with the multiplexer. This is the section that monitors the modem and controls the functions of the logic. The timing from the modem is used to perform the shifting and movement of data from buffer to buffer.

This is also the section that requests service from the line unit multiplexer. Service is requested when an assembled character is in the receive character buffer or when the transmit character buffer is empty. If the multiplexer grants the request, a character will either be removed from the Rec character buffer or a character will be loaded into the transmit character buffer.

A status register is present in each of the line units. These status registers are slaved to the master status register. The form of these status registers is effectively the same as the master status register without the octal line address bits.

II. The In/Out Registers with Transparent Mode Control

Both the transmit and receive side of each line unit has a buffer and a shift register. On the transmit side a character from memory is loaded into the transmit character buffer and at a later time is loaded into the transmit shift register. The character is shifted out of the shift register onto the transmit line one bit at a time by the transmit timing supplied from the modem. When the last bit of the character in the shift register has been transmitted, the character in the transmit character buffer is transferred into the transmit shift register and a service request

is generated. The computer has until the end of this character transmission to honor the request (160 microseconds for 50K baud data rate).

On the receive side of the line a character is assembled in the receive shift register by shifting in the incoming bits using the modem receive timing. When a complete character is in the register, it is transferred into the receive character buffer and a service request is generated. Also at this time the receive shift register is cleared in preparation for the first bit of the next character.

As each character is placed in a character buffer, it is looked at to determine if it is a control character. If a DLE occurs followed by a STX, the logic goes into a Transparent Mode.

If a DLE is found in the transmitted message while in the transparent mode, an additional DLE is generated by the hardware and added to the message. Word count overflow occurs at the end of the packet text to tell the transmit logic not to double any further DLE's. This allows correct transmission of the DLE, ETX, and 3 check sum characters at the end of the packet. After the check sum has been transmitted, the logic causes an interrupt to indicate that the packet transmission is complete.

If a DLE is found in the received message while in the transparent mode, the next character is examined to see if it is an ETX or a DLE. If it is a DLE it is not transferred to the receive character buffer and therefore only one DLE is stored in memory. In this way an even number of DLE's result in only every other DLE being stored and no mode change. If after an odd number of DLE's the next character is an ETX, the receiver is taken out of the

transparent mode and after receiving the check sum causes an interrupt to indicate that a packet has been received. SYN characters can be generated by the transmit side of the line unit, or the modem can apply them to the transmit data line. When receiving, the receive logic will see all the SYN characters, but will not pass any into memory except during transparent or check sum mode. By seeing the SYN characters the receive logic always remains in sync.

III. Modem Interface

This portion of the line unit monitors the status of the modem so this information can be passed on to the PDP-8I via the status register. The functioning of the modem, where applicable, can be controlled from the PDP-8I via the control register. In this logic is a section that can either take receive and send timing from the modem, or can generate the necessary timing to perform the shifting of bits or the movement of characters within the line unit.

1.6 Cyclic Check

The cyclic check will be performed on the data while it is residing in core not as it is being received. Only one-half adder/shift register will be used and it will access the data in core through the single-cycle data break facility of the PDP-8I. The hardware for the cyclic check will include the half adder/shift register, a current address register and a word count register.

The data break access for this unit is on a lower priority than the line multiplexer. The time between data requests will be adjusted to provide optimum system throughput.

A cyclic check on a block of data will be performed in the following manner: (a) the software will give to the hardware the starting address and size of the block to be treated and an indication of whether a check sum is to be generated or whether the block has a check sum and a test for data errors is required; (b) the device will then request single-cycle data breaks to access the block of data and on completion of the operation the device will do one of the following things: (1) If generating a check sum the device now requests three additional single-cycle data breaks to allow it to deposit the 24-bit check sum as three 8-bit characters into three locations in core contiguous with the block. A flag is then set to give an interrupt to indicate to the program that the operation is complete. (2) If checking a block for errors, a flag is set to request an interrupt to indicate to the program that the operation is complete and, in addition, an indication is given as to whether an error was detected or not (a zeroed shift register indicating no errors).

By performing the error control in this manner, only one hardware device is required per IMP instead of one per line. The programming of the cyclic check device and the required data break access are considered in the total system time study portion of this document.

It was found that the second polynomial in g(x) given in the RFQ should contain an x to the sixth (x^6) term to give the correct Bose-Chaudhuri code and to be consistent with the shift register shown.

1.7 HOST Interfaces

Three options for interfacing IMPs and HOSTs are provided (see Fig. 1.1).

Option 1 is the option being bid as part of this proposal, and is considered the preferred option. Under option 1 the HOST looks like another line unit to the IMP. Data transfer is serial. The HOST acts as though it were communicating with a modem.

The IMP side of the channel is a modified line unit running at a higher speed than the line units with modems. The IMP side also provides a modem simulator, making the IMP look like a standard modem to the HOST.

The advantage of option 1 is that most manufacturers provide interfaces with modems and that the IMP software interface can be standard.

Option 2 is to use the interfaces which DEC has developed for interfacing with a variety of other manufacturers' equipment, or variants of these interfaces.

Some of the other machines that Digital Equipment Corporation has interfaced to include:

IBM 360
IBM 7040, 7044 and 7090
UNIVAC 1108
SDS 940
CDC 3600.

These interfaces are, for the most part, parallel-to-parallel and either direct memory-to-memory or bus (PDP) to Data Channel type interfaces.

The third available option is to place the responsibility upon the HOST staff to build an interface to the PDP-8I bus. Many HOST staffs are expert in electronics, and are familiar with DEC modules and DEC hardware. They may well elect under option 3 to build their own interface. If they desire, we would be pleased to provide consulting services under separate contract.

Connecting more than one HOST to a single IMP can be done readily if options 1 or 2 are used. For option 1, the only additional hardware needed would be another modified line unit for each additional HOST. For option 2, the interprocessor buffers run independently of each other and will be multiplexed by the DM01 multiplexer.

Chapter 2
Software

2.1 Core Allocation

The IMP software is divided into three modules: carrier-side software, HOST-side software, and the buffer pool. Each module is assigned to a separate core field of 4096 words.

In regard to module 2 (buffer pool), it is seen from the analysis in Chapter 3 that the present core allocation is adequate.

In regard to module 1 (HOST-side software), standard IMP-HOST interface software will be provided to fit in one field with some room to spare. There is uncertainty, however, regarding the space required by the HOST's programming staff for modification to our standard software. If the required modification at some installation is so large as to exceed the spare space, then additional 4K fields will be required for module 1 at that installation.

In regard to module 0 (carrier-side software) space will be provided for keeping statistics on network performance. It is proposed that when this space overflows, the accumulated old statistics be dumped to a selected HOST having responsibility for keeping track of network performance*. If this proposed scheme is unacceptable, then either (1) the amount of statistics kept in IMPs will be limited or (2) additional core fields for module 0 will be required or (3) auxiliary storage such as disk will be required.

---

* For example, UCLA, as suggested in the ARPA statement of work.

The proposed modular layout has the advantage of lending itself easily to later growth of the system; little change in the software (in some cases no change) is required if additional fields are added. The modular layout also lends itself to a simple memory protection scheme (see Section 2.3).

2.2 Buffer Pool

A buffer-pool scheme is proposed. In this scheme, a set of buffers is made available in a common pool. Any program requiring a buffer is assigned the next available free buffer. When the buffer becomes free again, it is released to the common pool.

The contents of buffers are never moved. All transmission of data in buffers is achieved by changing pointers to buffers.

All buffers are 130 words in length. There are 31 buffers. As will be seen below, we place an entire packet, including the initial DLE STX, into a buffer, one character to a word. Thus, packets of up to 1024 bits, starting with the first character of the header, can be accommodated in one buffer*.

2.3 Memory Protection

Since the HOST system programmers have access to the HOST-side module, it is possible that undebugged programs will occasionally

---

* If it were acceptable to shorten the maximum packet length by two characters, then the buffers would be 128 words long, and there would be 32 buffers.

be run in the IMP. Protection of the IMP software against such bugs in field 1 is obtained by restricting the HOST-side programs from storing into field 0. (HOST-side programs may store into fields 1 and 2.)

The restriction is imposed by causing an interrupt whenever a CDF (change data field) instruction is given from a field other than 0. The interrupt is serviced by the field 0 programs, which prohibit the data field to be changed to field 0.

Because of the modular layout, this protection scheme would be easy to implement if desired. It does not, however, form a part of the present proposal.

2.4 Packet and Message Formats and Protocol

A. Packet and Message Formats

The proposed packet format is as given in the ARPA statement of work, namely:

S	S	D	S		D	E	C	C	C	S	S
Y...Y		L	T	INFORMATION	L	T	S	S	S	Y...Y	
N	N	E	X		E	X	1	2	3	N	N

Characters are eight bits long. The ASCII definitions of the characters SYN, DLE, STX, and ETX are observed.

In the portion of the packet labeled INFORMATION arbitrary 8-bit binary characters are allowed. Any DLE encountered on transmission

is automatically doubled up by the hardware if and only if it is contained within INFORMATION (see Section 1.5). This procedure guarantees that, on the line, an ETX in INFORMATION is never preceded by an odd number of DLEs. In the entire packet, then, the only ETX on the line preceded by an odd number of DLEs is the ETX following the DLE following INFORMATION. By sensing for this condition, the receiving hardware detects that INFORMATION has terminated. During reception, the hardware deletes the second [redacted] of adjacent DLEs encountered in INFORMATION, thus restoring the entire packet to the precise format transmitted (Section 1.5).

The characters CS1, CS2, and CS3 represent the cyclic parity check, as defined in the ARPA work statement (Section 1.6).

Within INFORMATION, the packet is divided into a HEADER, followed by TEXT. The HEADER is six characters long. The TEXT is 0 to 117 characters long. Thus, INFORMATION is 6 to 123 characters long.

The TEXT may contain ASCII code or some other character code or binary information. The TEXT is always treated by the IMP as pure 8-bit binary.

The HEADER contains the following 48 bits of information, packed into 6 characters.

1. Destination code (8 bits). See 2, below.
2. Origin code (8 bits). It is suggested that each IMP and each HOST in the network have a separate code, which is used in the destination and origin characters of the HEADER. In this way, we would allow for the possibility that an IMP may send a message to

a HOST other than its own*. For example, we envision the possibility that IMPs may dump statistical information to a selected HOST by means of IMP-generated messages.

3. Message ID (16 bits).

4. Packet number (5 bits).

5. Handover number (6 bits). The handover number is set to 0 by the originating IMP. Each time the packet is relayed by an IMP, this number is incremented by 1. In this fashion, cycles in the routing may be detected. However, if the type of routing doctrine proposed in Section 3.5 is adopted, cycles become impossible and the handover number may be unnecessary (see Section 2.5 for another possible use for these 6 bits).

6. Packet priority (1 bit).

7. End-to-end message acknowledgement required (1 bit).

8. Last packet in message (1 bit).

9. TEXT is for IMP (1 bit). This bit will not be needed if suggestion in 2, above, is adopted.

10. TEXT is binary.

The preceding defines the packet format. Message format is defined by the individual HOST for the purpose of communicating with its own IMP. However, we define a standard message format which will be implemented for check-out purposes; it will also be available for all installations that care to use it. Others are free to modify it, or to provide their own.

The standard message format consists of a 6-character HEADER followed by 1 to 8 TEXTBLOCKS. Each TEXTBLOCK is 0 to 117 characters long.

---

* If codes are assigned only to HOSTS, a receiving HOST would not know whether a message came from a HOST or an IMP.

The message HEADER format is identical to the packet HEADER format. The TEXTBLOCK corresponds to a packet TEXT.

A message may be composed out of 1 to 8 packets; conversely, a message may be decomposed into 1 to 8 packets. The process of composing a message from packets is as follows: The HEADER of packet 1 becomes the HEADER of the message; the TEXT of packet 1 becomes TEXTBLOCK 1; the TEXT of packet 2, if any, becomes TEXTBLOCK 2; ...; the TEXT of packet 8, if any, becomes TEXTBLOCK 8. The TEXTBLOCKs are concatenated. No character is given special consideration; there is no cyclic check.

On reception by the HOST, the message HEADER contains some unnecessary or meaningless information. It is easier, however, to leave it in than to take it out. On transmission from the HOST, this unnecessary or meaningless information is left blank by the HOST. The required information is provided by the IMP when the IMP decomposes the message into packets.

B. Communication Protocol

Protocol deals with the overall procedure for exchanging packets and messages. Included are such questions as how packets are acknowledged, how retransmitted if not acknowledged, how to proceed if buffers become full, etc.

Designing good communication protocols is, as we have found from considerable experience in the matter, a difficult, time-consuming, and subtle undertaking; it will be taken up as an important task in the design phase of the proposed contract. No protocol design is attempted in the present proposal. Instead, the matter is discussed, and a few considerations are outlined.